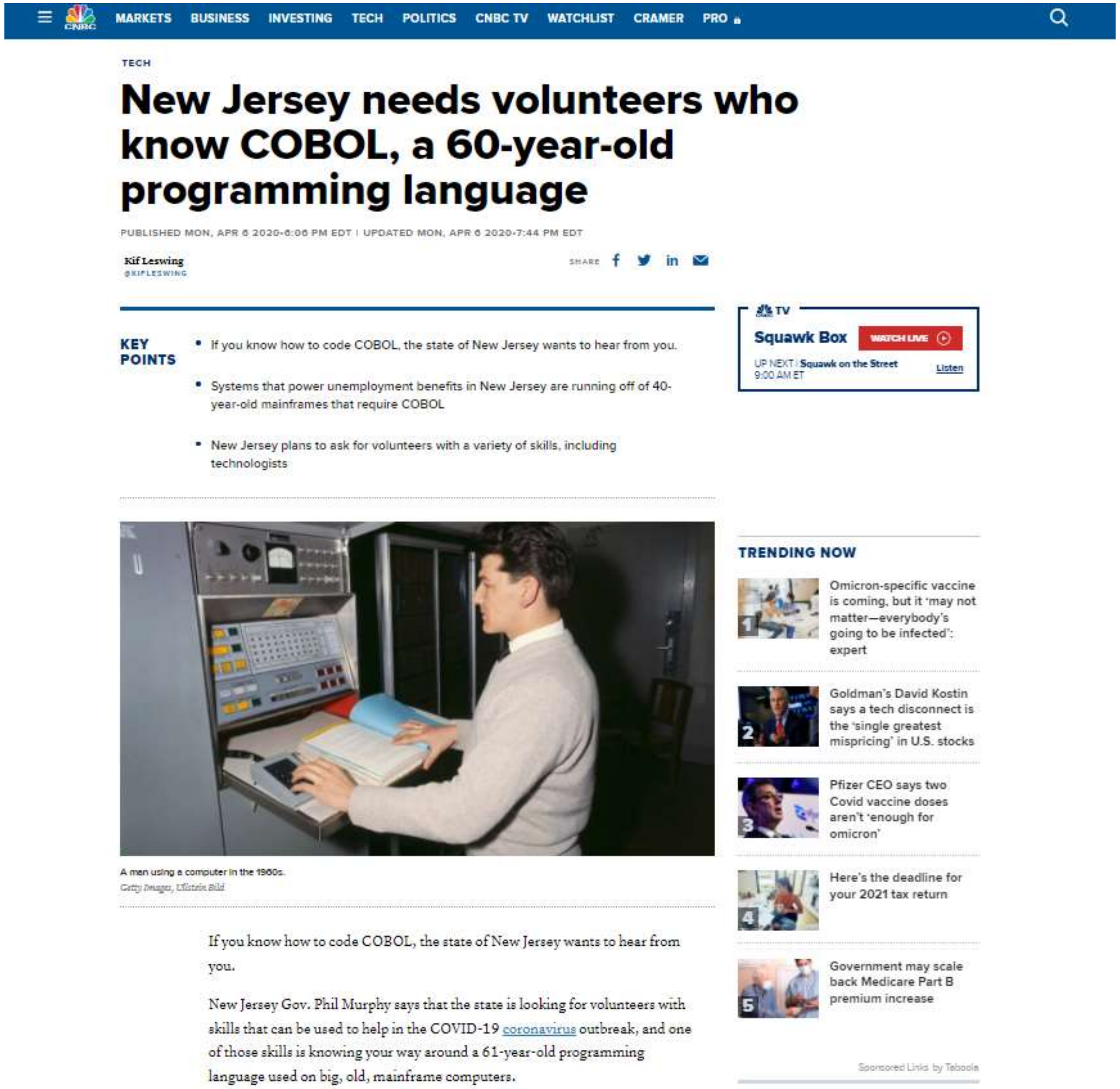
WEBSERVICES

History


MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV WATCHLIST CRAMER PRO

TECH

New Jersey needs volunteers who know COBOL, a 60-year-old programming language

PUBLISHED MON, APR 6 2020-6:06 PM EDT | UPDATED MON, APR 6 2020-7:44 PM EDT

Kif Leswing
@KIFLESWING

KEY POINTS

- If you know how to code COBOL, the state of New Jersey wants to hear from you.
- Systems that power unemployment benefits in New Jersey are running off of 40-year-old mainframes that require COBOL
- New Jersey plans to ask for volunteers with a variety of skills, including technologists

A man using a computer in the 1960s.
Getty Images, Ullstein Bild

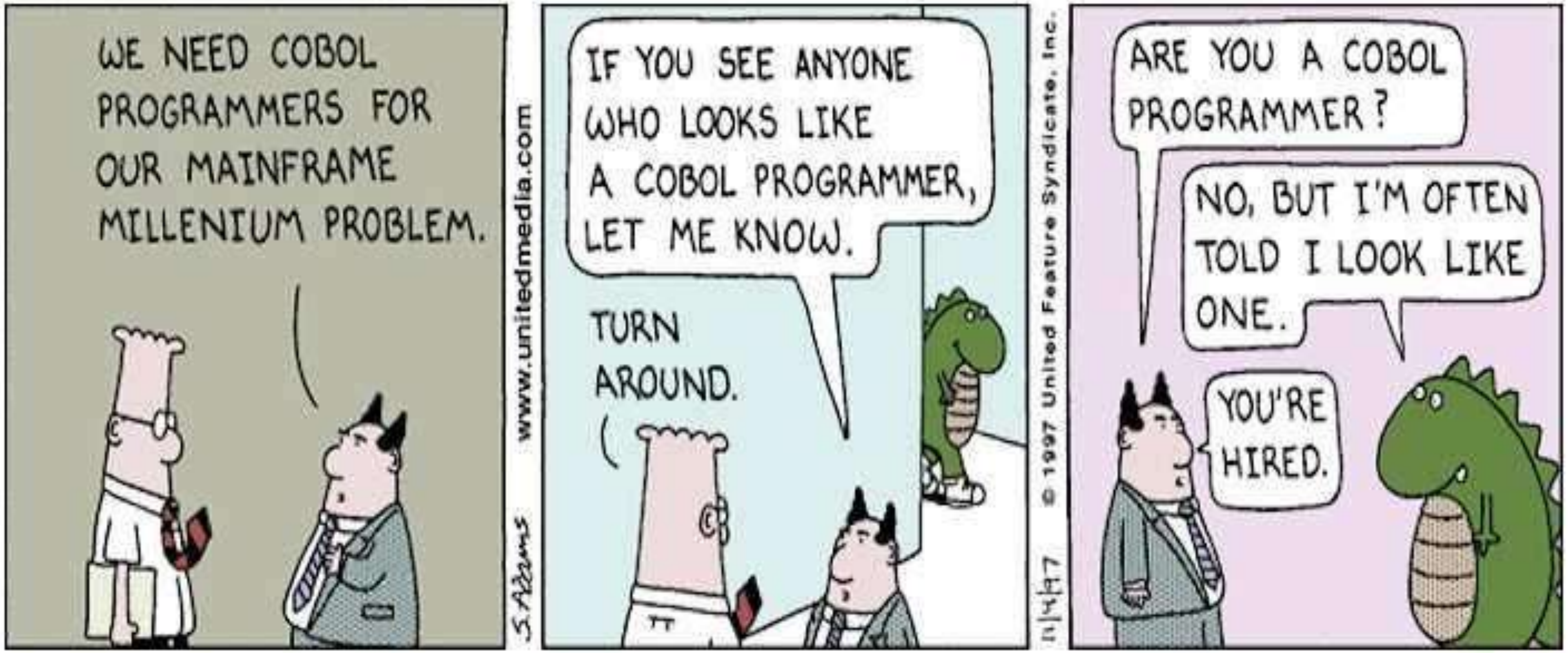
If you know how to code COBOL, the state of New Jersey wants to hear from you.

New Jersey Gov. Phil Murphy says that the state is looking for volunteers with skills that can be used to help in the COVID-19 coronavirus outbreak, and one of those skills is knowing your way around a 61-year-old programming language used on big, old, mainframe computers.

TRENDING NOW

1. Omicron-specific vaccine is coming, but it 'may not matter—everybody's going to be infected': expert
2. Goldman's David Kostin says a tech disconnect is the 'single greatest mispricing' in U.S. stocks
3. Pfizer CEO says two Covid vaccine doses aren't 'enough for omicron'
4. Here's the deadline for your 2021 tax return
5. Government may scale back Medicare Part B premium increase

Sponsored Links by Taboola


WE NEED COBOL PROGRAMMERS FOR OUR MAINFRAME MILLENIUM PROBLEM.
www.unitedmedia.com
S. Adams
IF YOU SEE ANYONE WHO LOOKS LIKE A COBOL PROGRAMMER, LET ME KNOW.
TURN AROUND.
© 1997 United Feature Syndicate, Inc.
11/4/97
ARE YOU A COBOL PROGRAMMER?
NO, BUT I'M OFTEN TOLD I LOOK LIKE ONE.
YOU'RE HIRED.

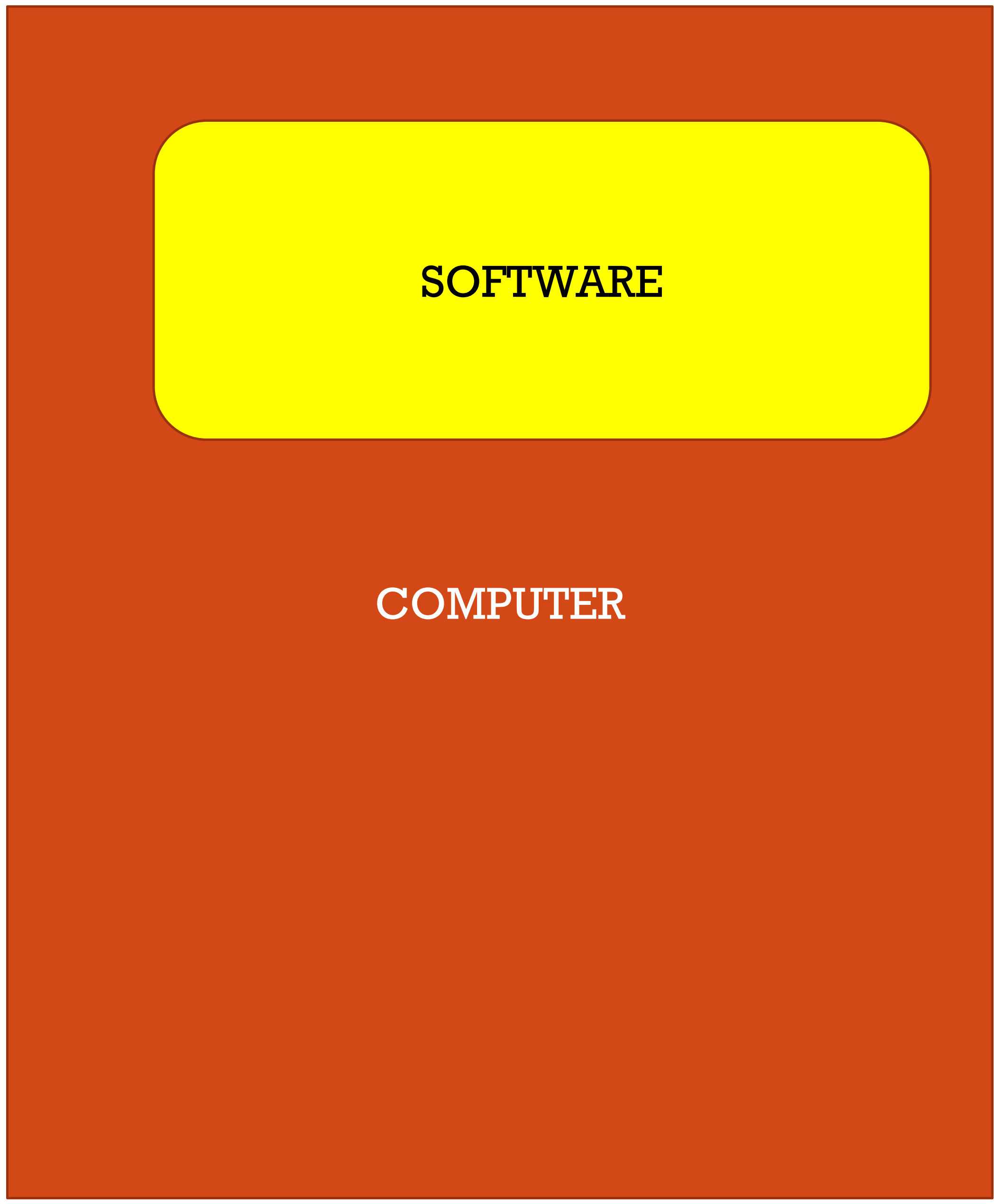
COMPONENT ARCHITECTURES

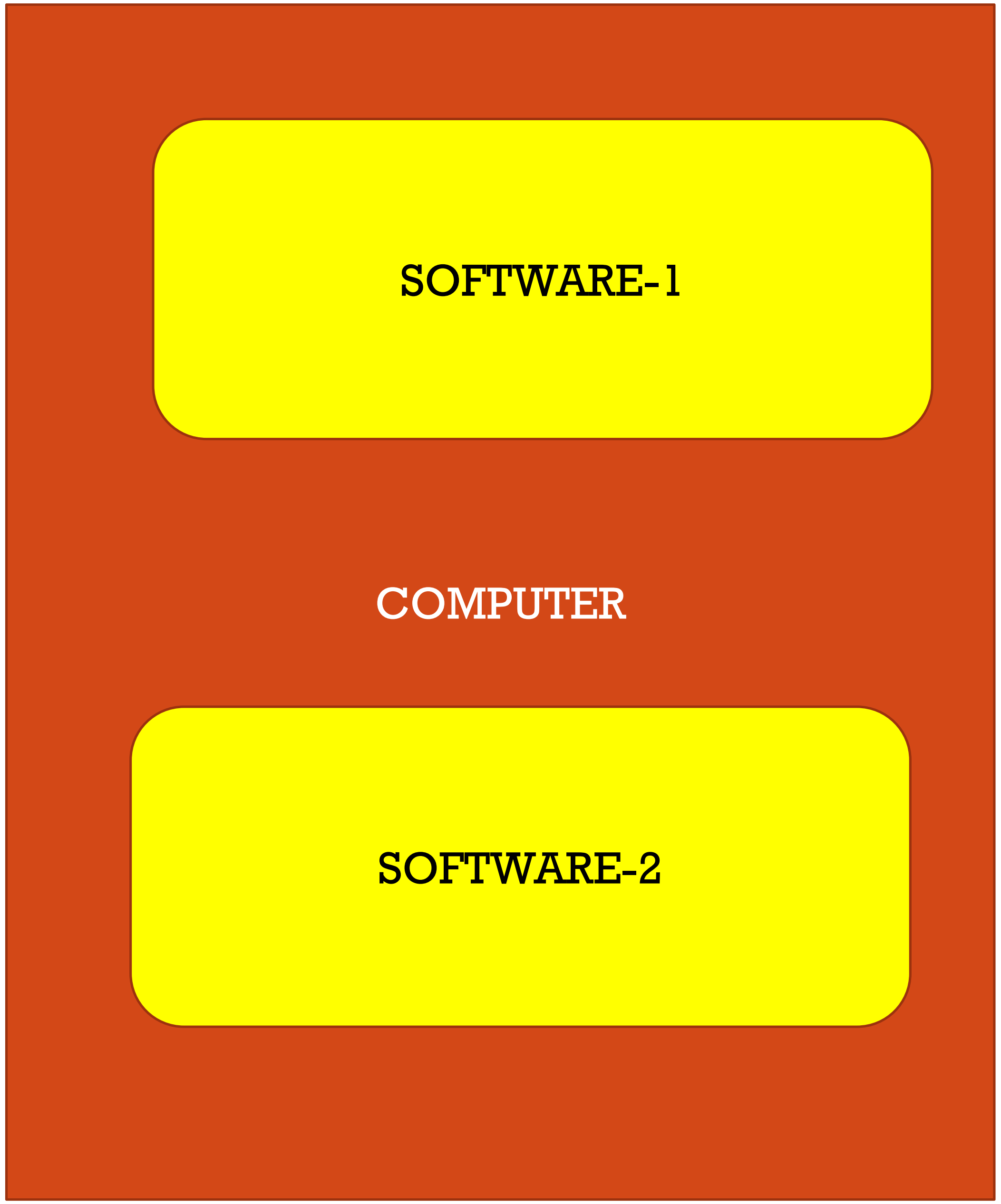
- A component architecture is a method of designing software components
  - The components should be easily connected together, reused, or replaced
  - Without re-compiling the application that uses them.
- A Component, is a piece of software that:
  - Is like a library, rather than a stand-alone application
  - Is distributed in a compiled, executable form
  - Exposes a group of methods and properties to its client application

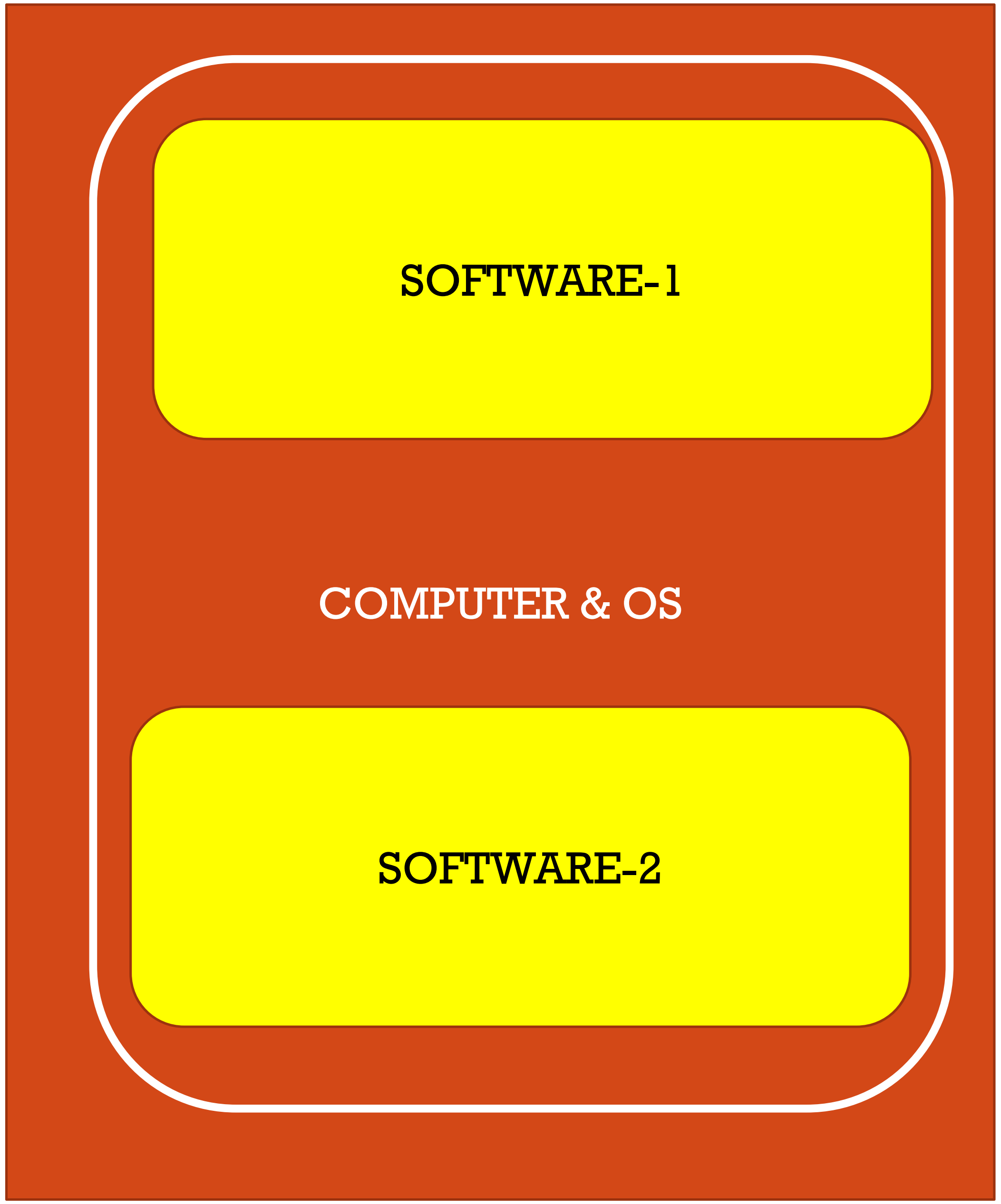
HISTORICAL APPROACHES

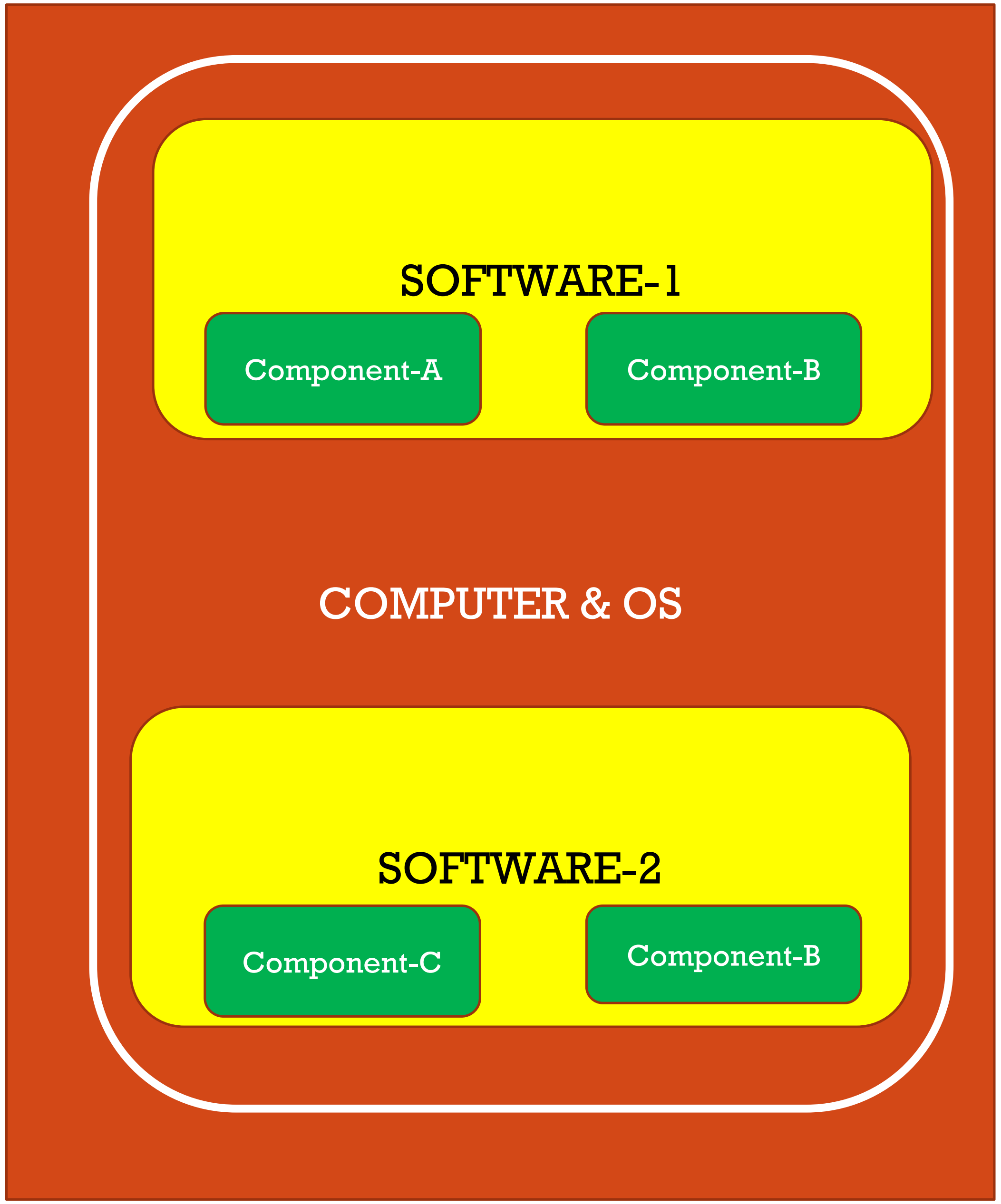
1. CORBA (OMG)
2. DCOM (Microsoft)
3. XPCOM (Mozilla)
4. RMI (Sun Microsystems) on JRMP/IIOP

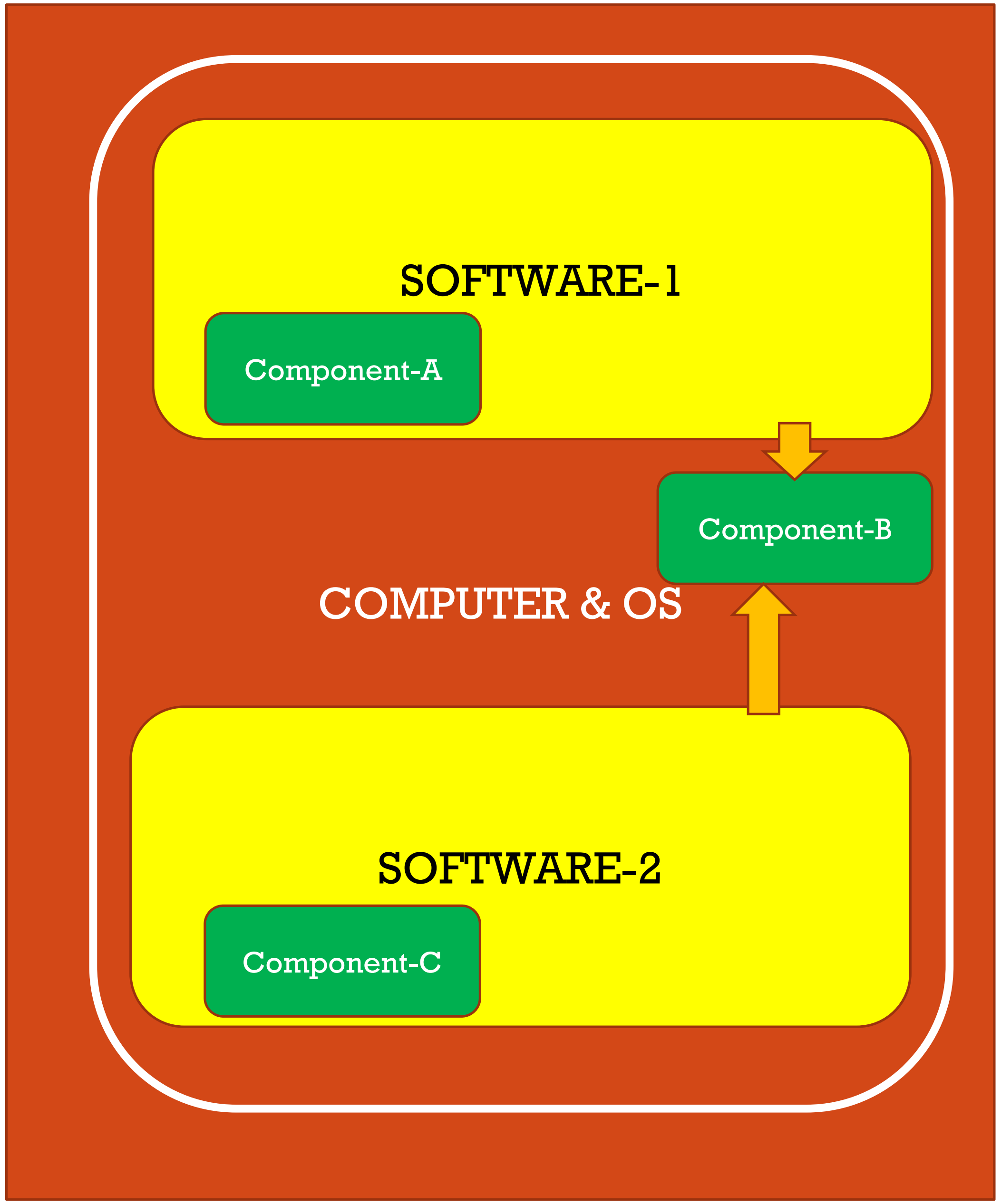
COMPUTER

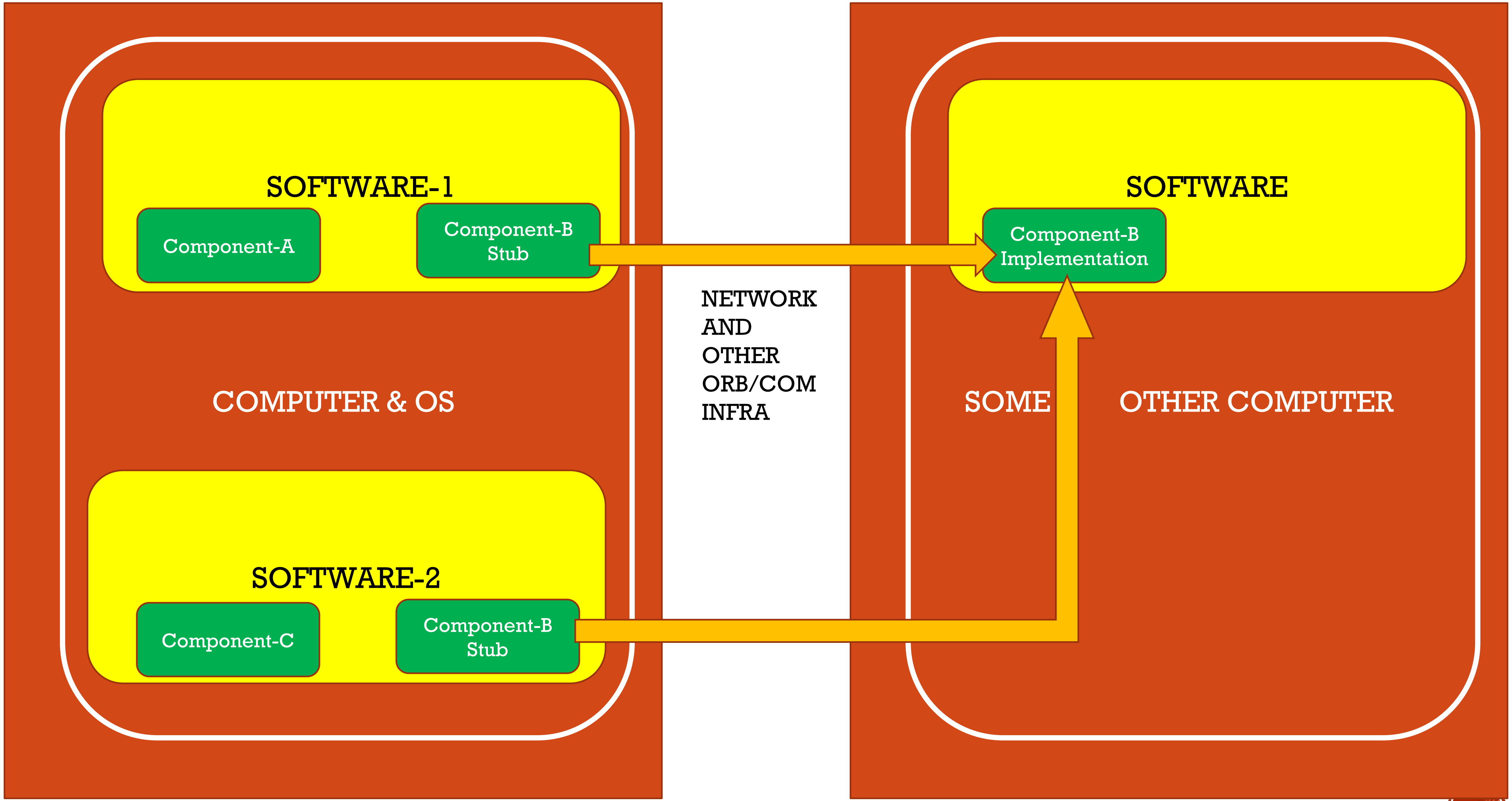

SOFTWARE
COMPUTER


SOFTWARE-1
COMPUTER
SOFTWARE-2


SOFTWARE-1
COMPUTER & OS
SOFTWARE-2


SOFTWARE-1
Component-A
Component-B
COMPUTER & OS
SOFTWARE-2
Component-C
Component-B

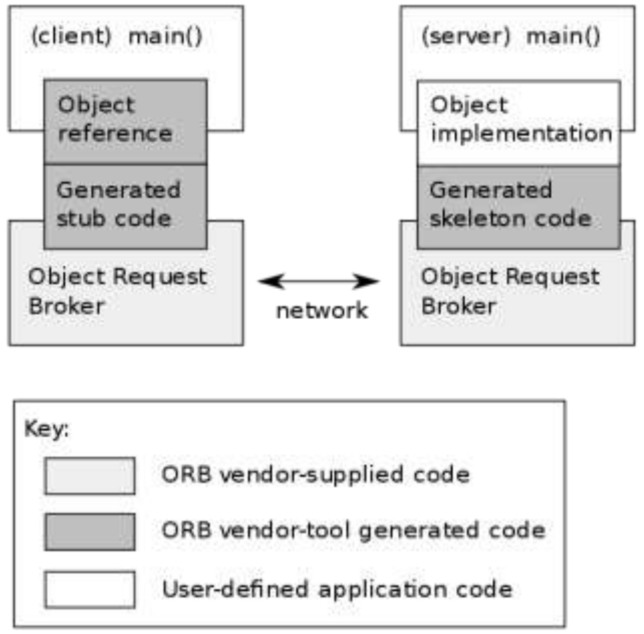

SOFTWARE-1
Component-A
Component-B
COMPUTER & OS
SOFTWARE-2
Component-C

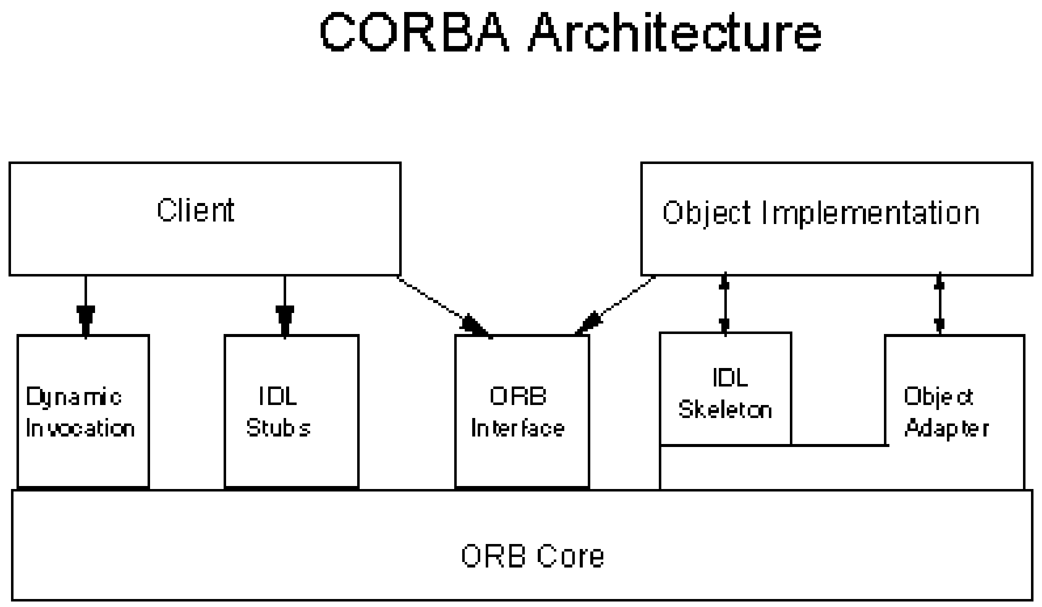


DESCRIBE DISCOVER COMMUNICATE

CORBA

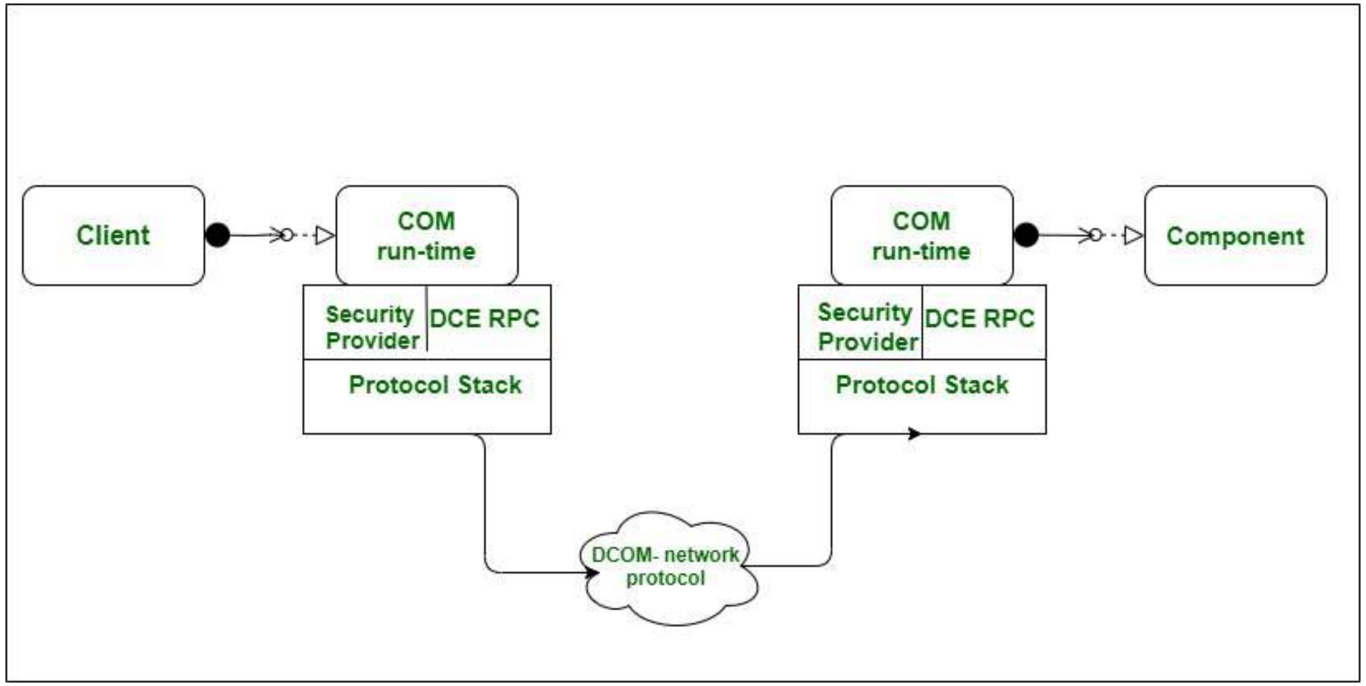
- CORBA (OMG) It is standards-based, vendor-neutral, and language-agnostic. Very powerful but limited however by its complicated way of utilizing the power and flexibility of the Internet.





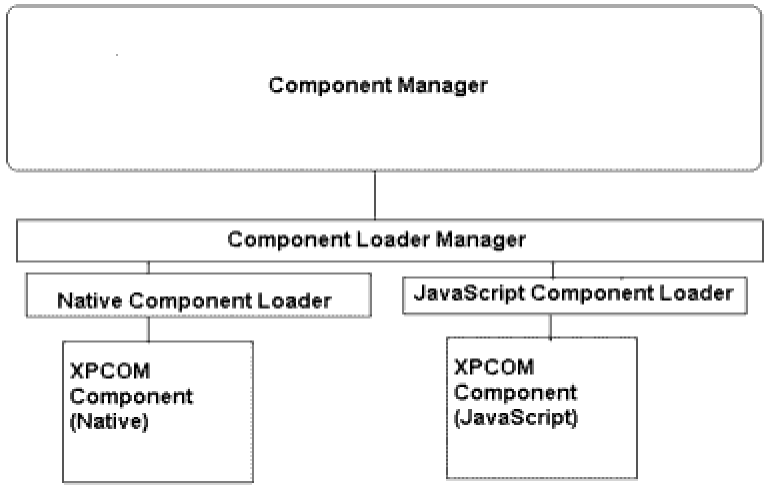
DCOM

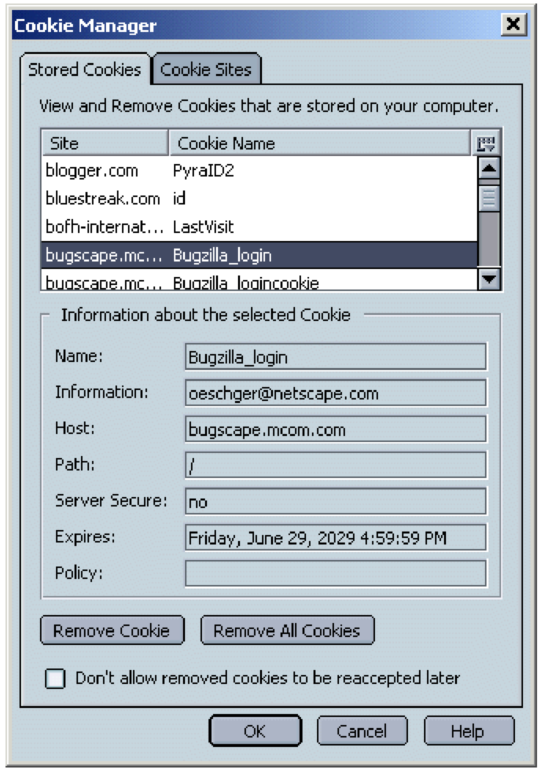
- DCOM (Microsoft) Distributed Computing platform closely tied to Microsoft component efforts such as OLE, COM and ActiveX.



XPCOM

- XPCOM is a Mozilla made competitor to DCOM (Microsoft). Stands for Cross.





View and Remove Cookies that are stored on your computer.

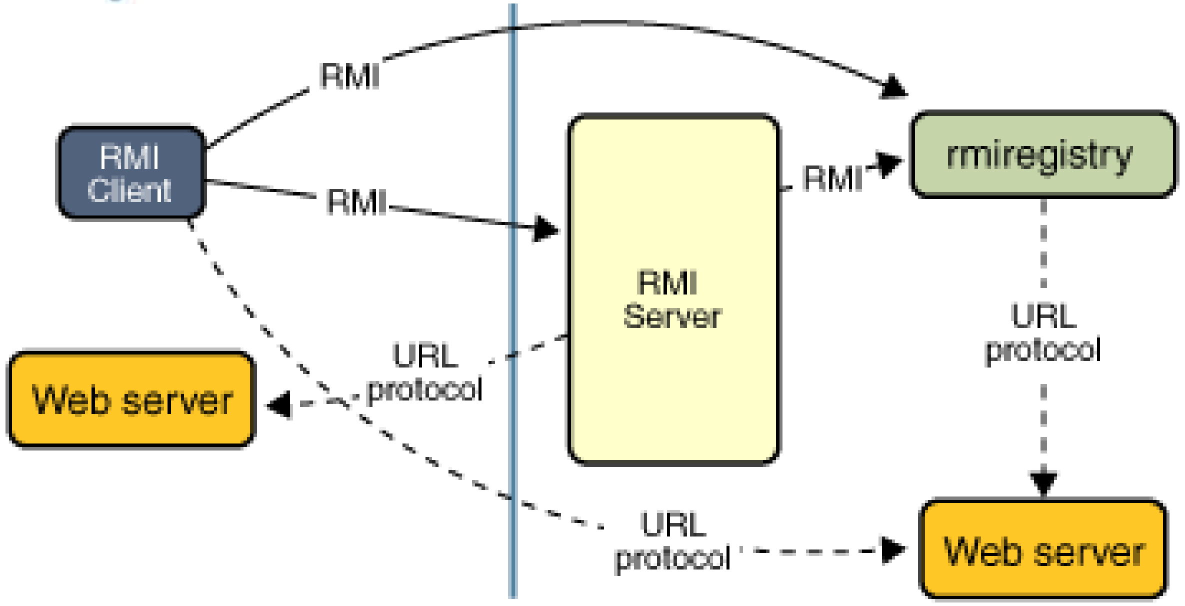
Site	Cookie Name
blogger.com	PyraID2
bluestreak.com	id
bofh-internat...	LastVisit
bugscape.mc...	Bugzilla_login
bugscape.mc...	Bugzilla_logincookie

Information about the selected Cookie

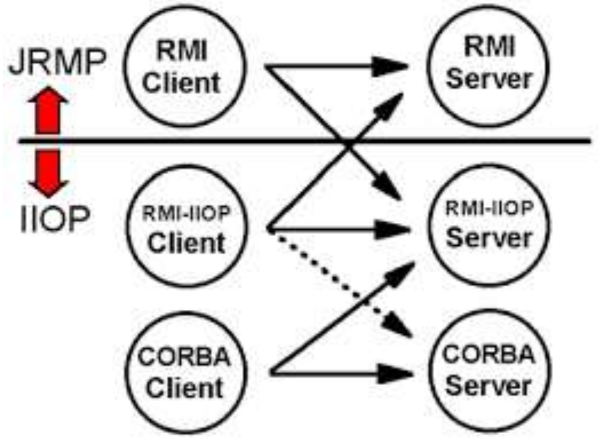
Name:	Bugzilla_login
Information:	oeschger@netscape.com
Host:	bugscape.mcom.com
Path:	/
Server Secure:	no
Expires:	Friday, June 29, 2029 4:59:59 PM
Policy:	

☐ Don't allow removed cookies to be reaccepted later

JAVA - RMI



- Java RMI provides the mechanism by which the server and the client communicate and pass information back and forth. This build a distributed object application.
- The RMI over IIOP implementation supports interop with CORBA



HISTORICAL APPROACHES (MORE TO READ)

- IDL: https://www.omg.org/spec/IDL/About-IDL/
  - Example: http://jmvidal.cse.sc.edu/csce590/spring02/corba-idl-intro.pdf
- DCOM (Microsoft) :
  - (COM) https://www.cs.umd.edu/~pugh/com/
  - https://condor.depaul.edu/elliott/513/projects-archive/DS420Fall98/Edinburgh/dcom.htm
  - XPCOM (Mozilla) : https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM
- CORBA (OMG) It is standards-based, vendor-neutral, and language-agnostic. Very powerful but limited however by its complicated way of utilizing the power and flexibility of the Internet. : https://www.omg.org/spec/CORBA/About-CORBA/
- RMI (Sun Microsystems) https://docs.oracle.com/javase/tutorial/rmi/overview.html
- Web Services (W3C) Web services **are more of an evolution than a revolution**

WHAT IS A WEBSERVICE?

Definition: A Web Service is a standards-based, language-agnostic software entity, that accepts specially formatted requests from other software entities on remote machines via vendor and transport neutral communication protocols, producing application specific responses.

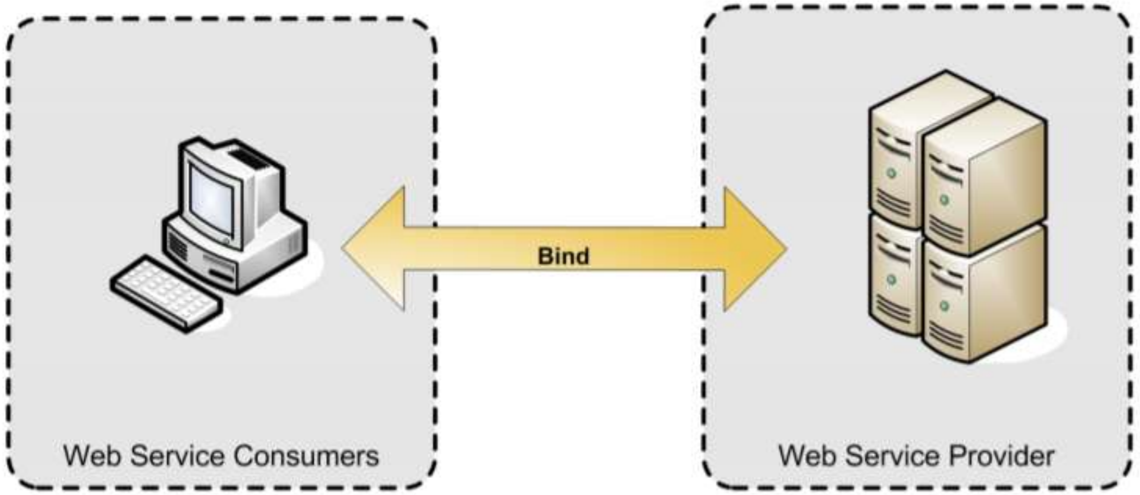
- Standards based - W3C, etc
- Language agnostic - Any programming language
- Formatted requests - XML, JSON, EDN, BSON etc
- Remote machines – Over the network (web)
- Vendor neutral – No one vendor controls the standards or specifications
- Transport neutral – HTTP, SMTP, ??
- Application specific responses – *Business-aware* responses

BENEFITS OF WEBSERVICES

- Loosely Coupled
  - Each service exists independently of the other services that make up the application.
  - Individual pieces of the application to be modified without impacting unrelated areas.
- Ease of Integration
  - Data is isolated between applications creating ’silos’.
  - Web Services act as glue between these and enable easier communications within and across organisations.
- Service Reuse
  - Takes code-reuse a step further and reduces duplication
  - (Theoretically) A specific function within the domain is only ever coded once and used over and over again by consuming applications.

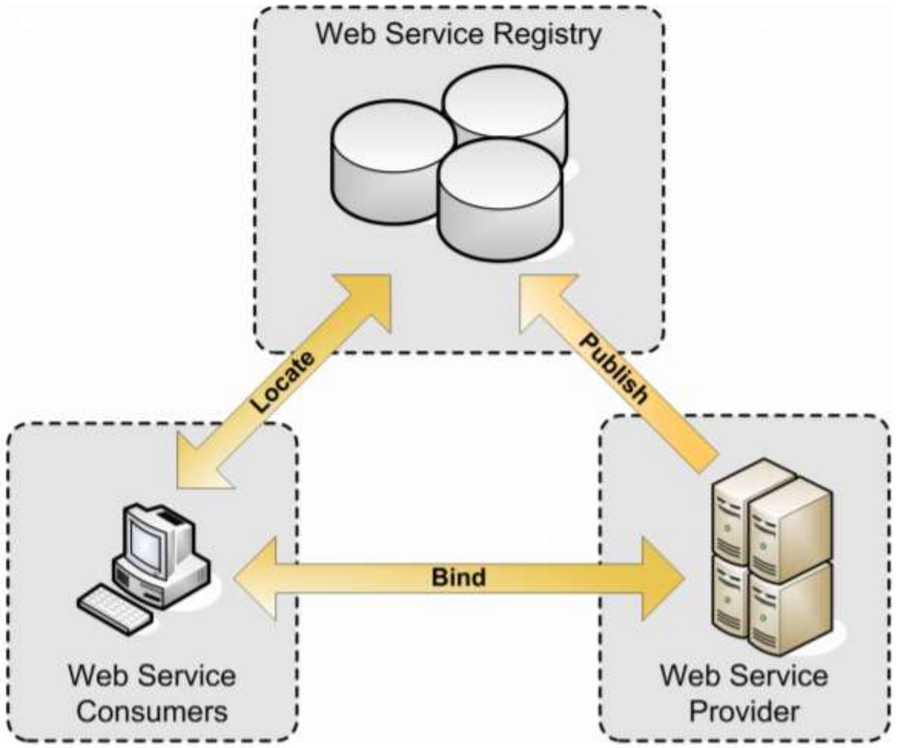
WEBSERVICES ARCHITECTURE (SIMPLE)

- The simplest Web service system has two participants:
  - A service producer (provider)
  - A service consumer (requester).
- The provider presents the interface and implementation of the service
- The requester uses the Web service.



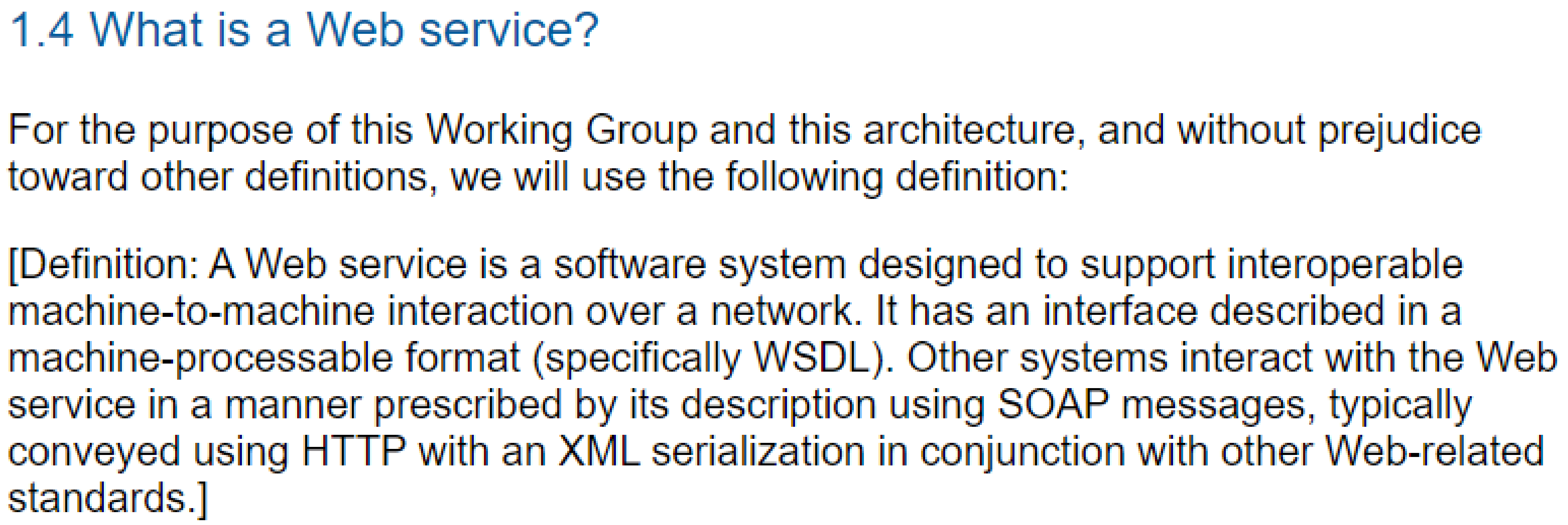
WEBSERVICES ARCHITECTURE (COMPLEX)

- A registry, acts as a broker for Web services.
- A provider, can publish services to the registry
- A consumer, can then discover services in the registry and start using it.



WEBSERVICES (DECADE OLD DEFINITION)

- https://www.w3.org/TR/ws-arch/#technology



1.4 What is a Web service?

For the purpose of this Working Group and this architecture, and without prejudice toward other definitions, we will use the following definition:

[Definition: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.]

WEBSERVICES

- DEFINITION (WSDL)
- DISCOVERY (UDDI)
- COMMUNICATION (SOAP over HTTP)
- All focuses on a vendor-neutral widely agreed upon message-exchange format

WHAT IS XML?

- XML is a software- and hardware-independent tool for storing and transporting data.
- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML COMPONENTS

- Elements
  - The pairing of a start tag and an end tag.
- Attributes
  - A name-value pair that is part of a starting tag of an Element.
- Processing Instructions
  - Special directives to the application that will process the XML document.
- Comments
  - Messages helping a human reader understand the source code.
- Character
  - Data Characters (in a specific encoding) Entities Whitespace

Definition

The term **element** is a technical name for the pairing of a start tag and an end tag in an XML Document.

Production Rule

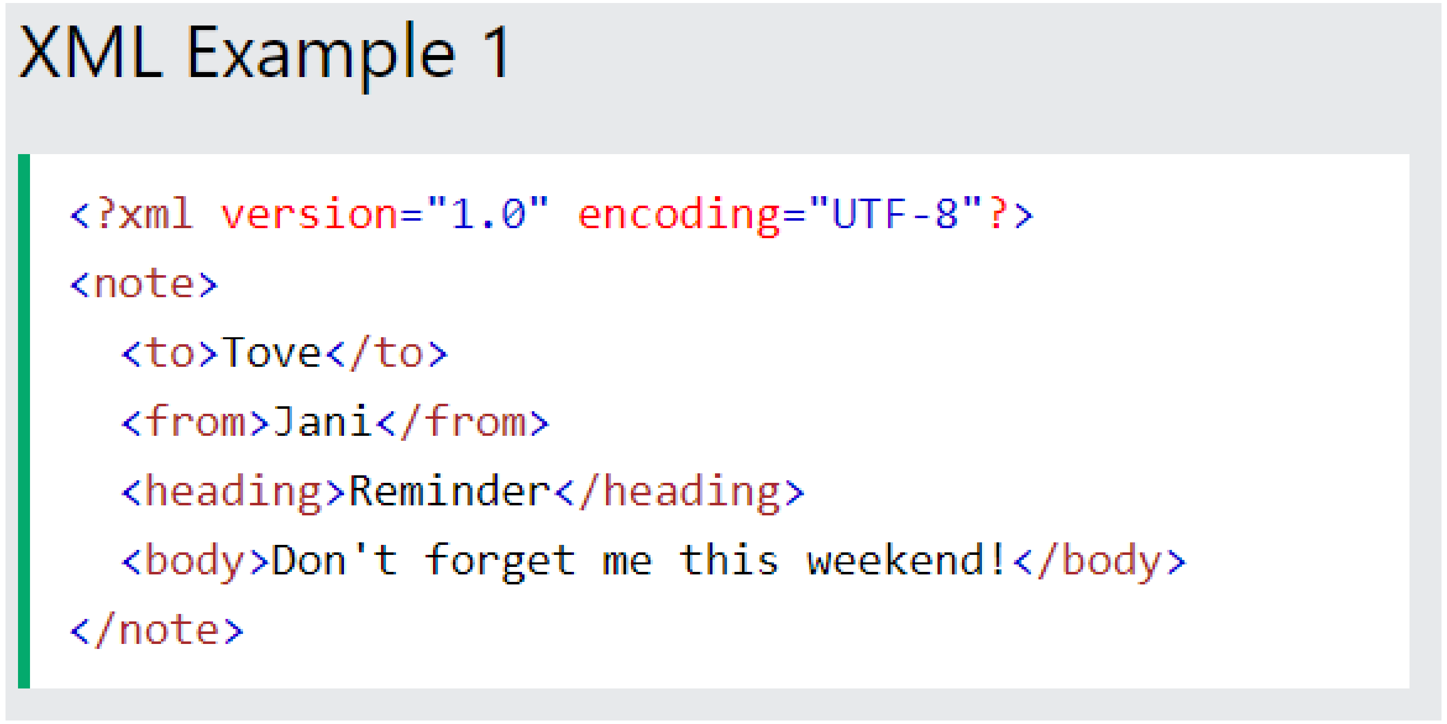
$$
\begin{array}{lcl}
\langle \textit{element} \rangle & ::= & \langle \textit{EmptyElement} \rangle \\
 & | & \langle \textit{STag} \rangle\ \langle \textit{content} \rangle\ \langle \textit{ETag} \rangle \\
\langle \textit{STag} \rangle & ::= & \text{`<'}\ \langle \textit{Name} \rangle\ \langle \textit{Attribute} \rangle^{\star}\ \text{`>'} \\
\langle \textit{ETag} \rangle & ::= & \text{`</'}\ \text{Name}\ \text{`>'} \\
\langle \textit{EmptyElement} \rangle & ::= & \text{`<'}\ \text{Name}\ \langle \textit{Attribute} \rangle^{\star}\ \text{`/>'}
\end{array}
$$

- XML Elements must be strictly nested!
- Element names can include letters, the underscore, hyphen and colon; they **must** begin with a letter.
- Element names are case sensitive!

XML EXAMPLE

- show real XML
  - Elements
  - Processing tags
  - Etc

W3Schools

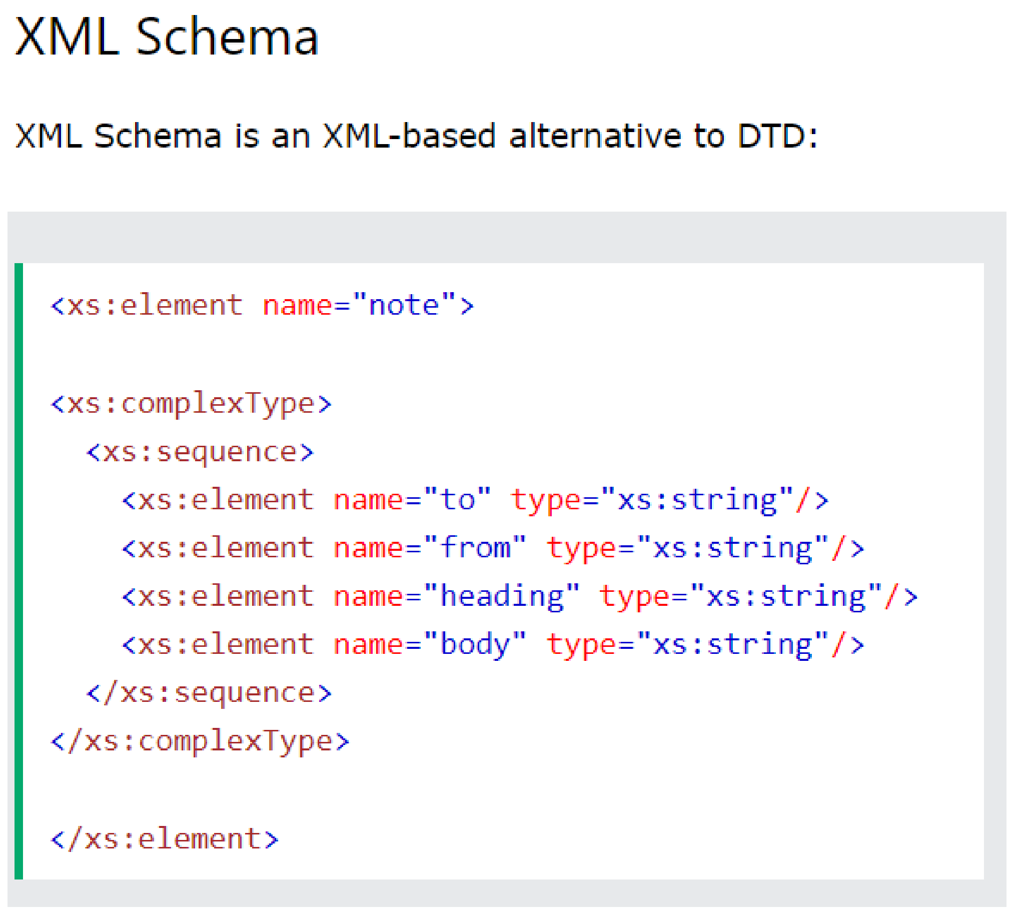


XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

SOME PROBLEMS AND FIXES

- Physical Structure of the document
  - Well formedness (Parsers)
- Logical Structure of the document
  - Validity (Schemas). Semantics of the elements?
- Element Name clashes between Documents
  - Namespaces
- Lets see examples……



XML Schema

XML Schema is an XML-based alternative to DTD:

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

TUTORIAL EXERCISE – XML & SCHEMA

- Use the question assigned to you in FSD-2 Lab Exam
- Create a Schema for Storing the data as XML (XSD File)
- Create an XML document as an example and validate using the schema

- Turn in the following
  1. XSD file
  2. XML file
  3. Screenshot of validation

SOAP AND WSDL WEBSERVICES

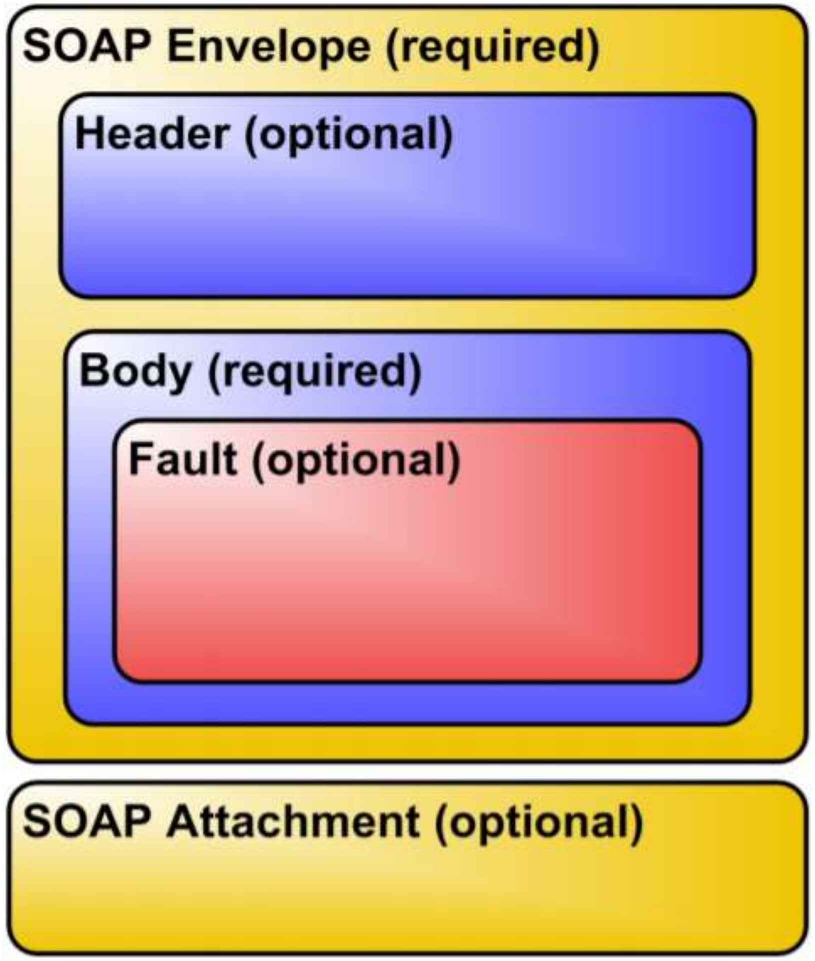
The Classical!

SIMPLE OBJECT ACCESS PROTOCOL

- SOAP is an industry accepted W3C specification for a ubiquitous XML distributed computing infrastructure.
  - A mechanism for defining the unit of communication.
  - A mechanism for error handling.
  - An extensibility mechanism Lives above the transport layer of OSI
  - Simply put its a mechanism that allows the transmission of XML documents, regardless of transport layer protocol.

STRUCTURE OF SOAP MESSAGES

- The root element of a SOAP message is the Envelope element.
- It contains an optional Header element and the required Body
- Elements called Faults can be used to describe exceptional situations.
- It can contain optional Attachments in MIME encoding for exchanging binary data.



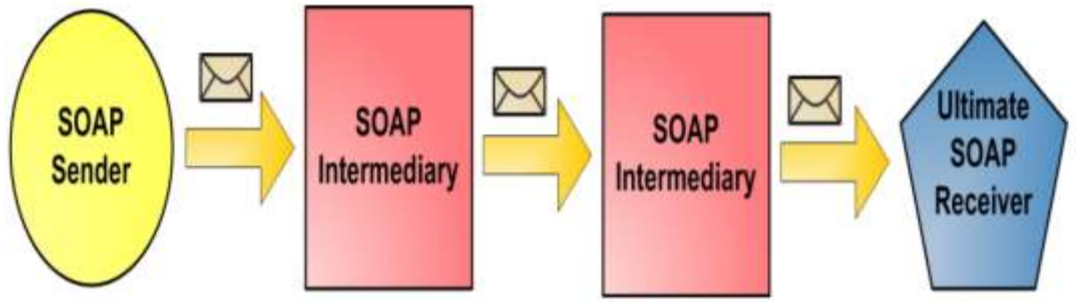
SOAP MESSAGES

- Lets see some examples!

W3Schools

SOAP MESSAGE DELIVERY

- The SOAP Sender creates and sends a SOAP Message to an ultimate SOAP Receiver.
- One or more optional SOAP Intermediaries can be positioned to intercept messages between the the sender and the receiver. They can perform filtering, logging, catching etc.
- The SOAP sender's intended destination is called the Ultimate SOAP Receiver.

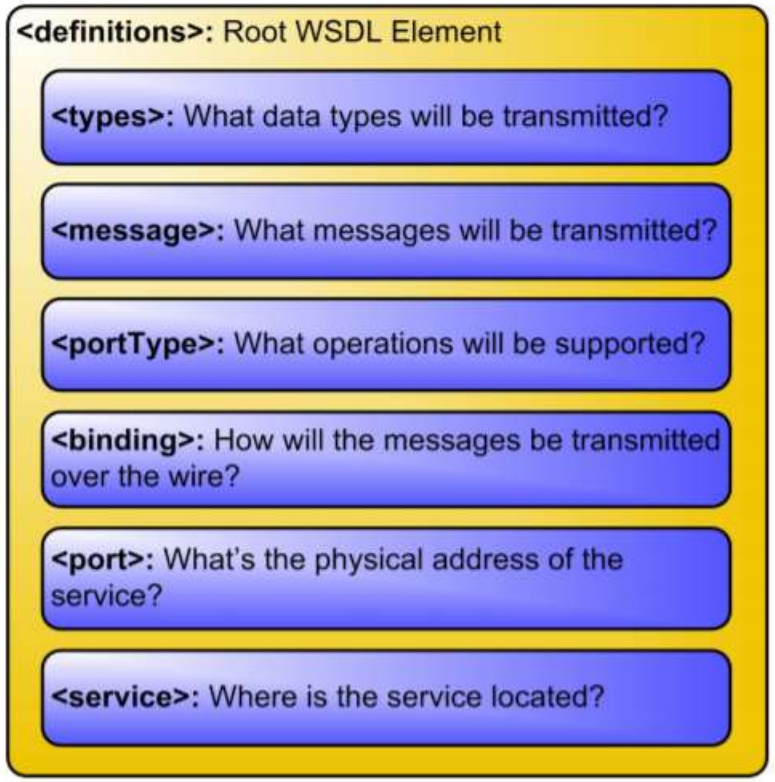


WEB SERVICES DEFINITION LANGUAGE

- Web Services Description Language (WSDL) is an XML format for describing all the information needed to invoke and communicate with a Web Service.
- It gives the answers to the questions Who? What? Where? Why? How?
- A service description has two major components:
  - Functional Description
    - Defines details of how the Web Service is invoked, where it's invoked.
    - Focuses on the details of the syntax of the message and how to configure the network protocols to deliver the message.
  - Nonfunctional Description
    - Provides other details that are secondary to the message (such as security policy) but instruct the requestor's runtime environment to include additional SOAP headers.

WSDL STRUCTURE

- A WSDL Document is a set of definitions with a single root element. Services can be defined using the following XML elements:
  - **Types,** think Data Type
  - **Message,** think Methods
  - **PortType,** think Interfaces
  - **Binding,** think Encoding Scheme
  - **Port,** think URL
  - **Service,** many URLs



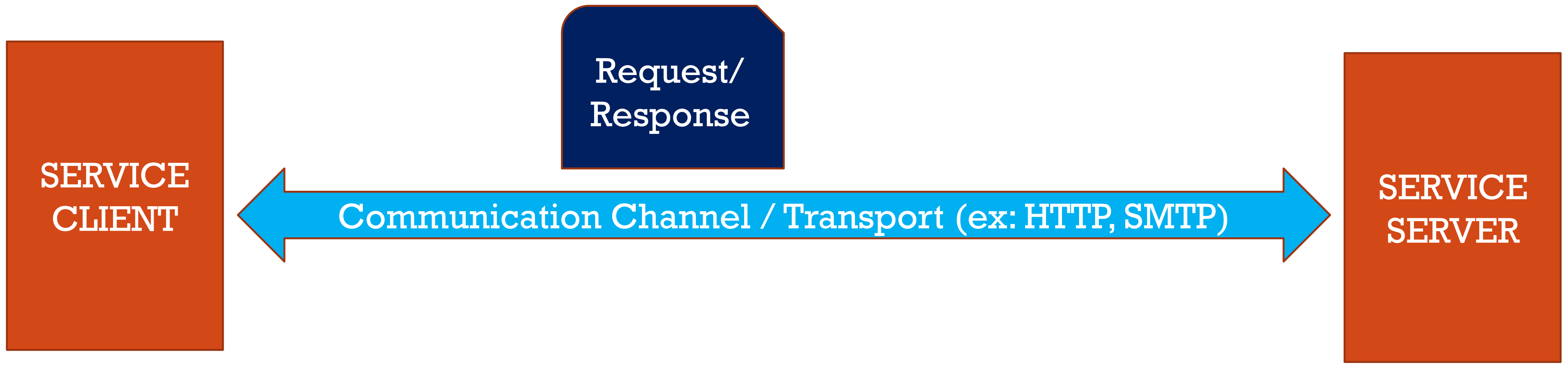
<definitions>: Root WSDL Element
<types>: What data types will be transmitted?
<message>: What messages will be transmitted?
<portType>: What operations will be supported?
<binding>: How will the messages be transmitted over the wire?
<port>: What's the physical address of the service?
<service>: Where is the service located?

WSDL EXAMPLE

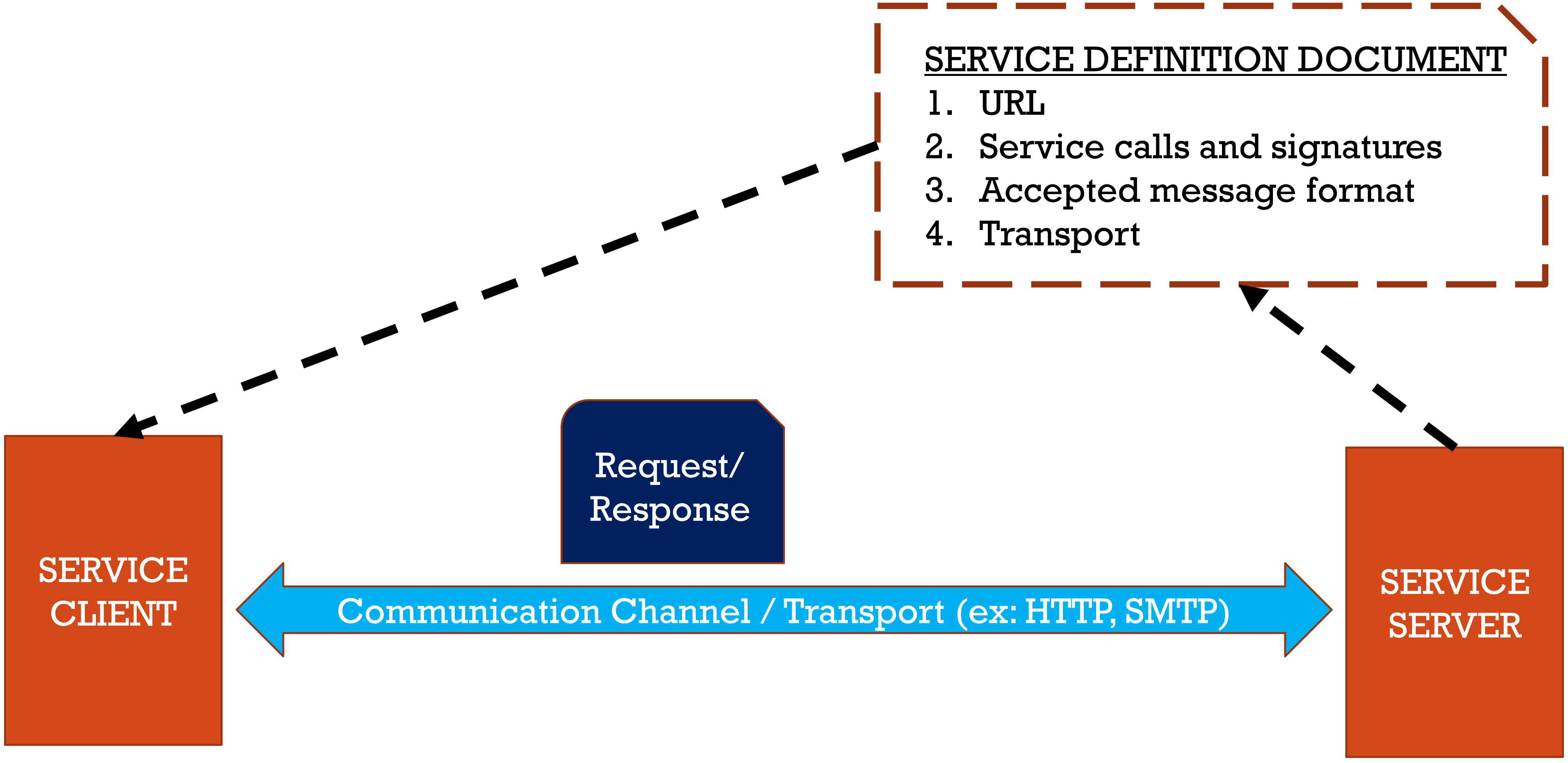
- Let's see some real examples!

  W3Schools

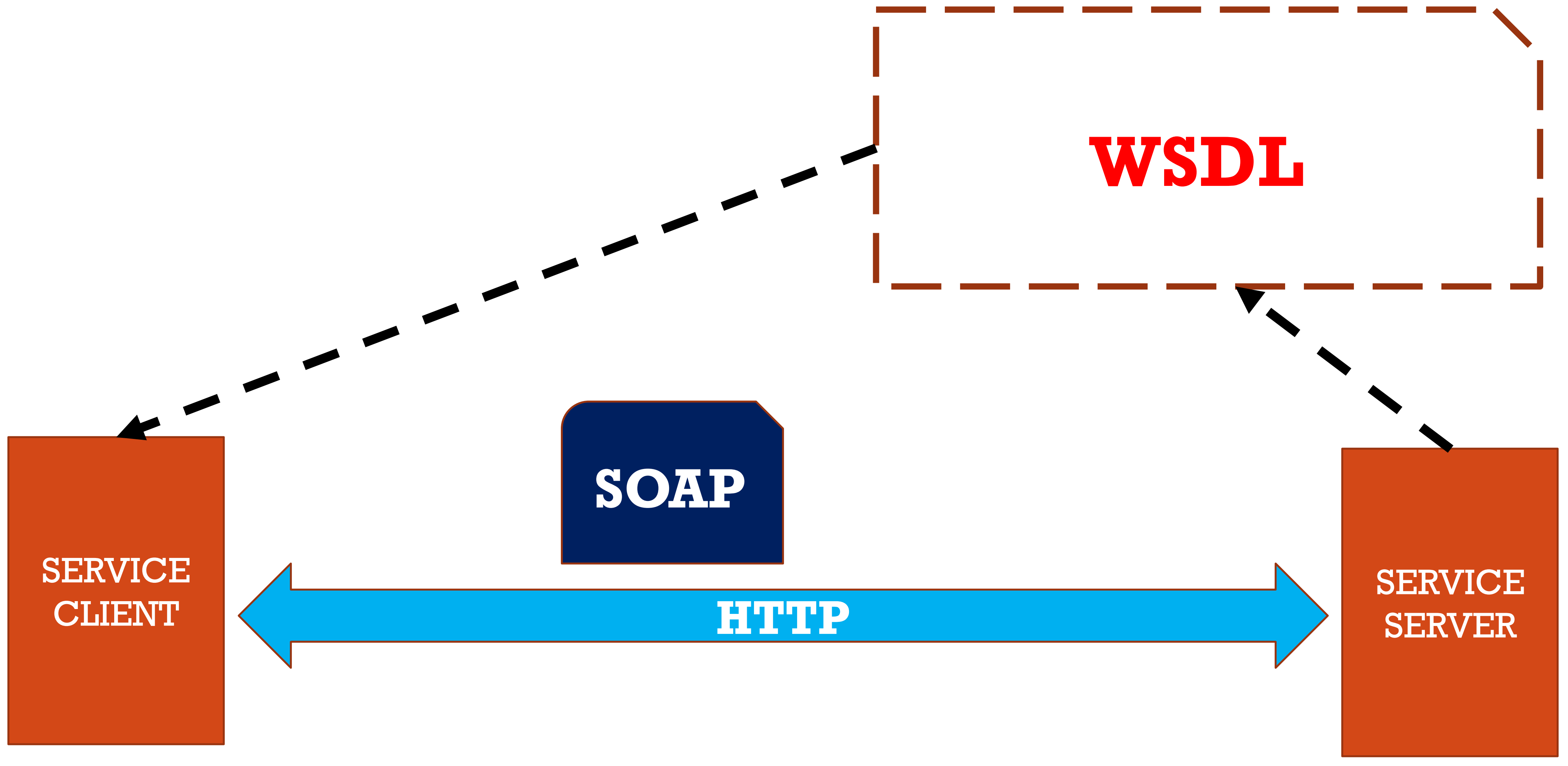
GENERAL ARCHITECTURE OF WEBSERVICES



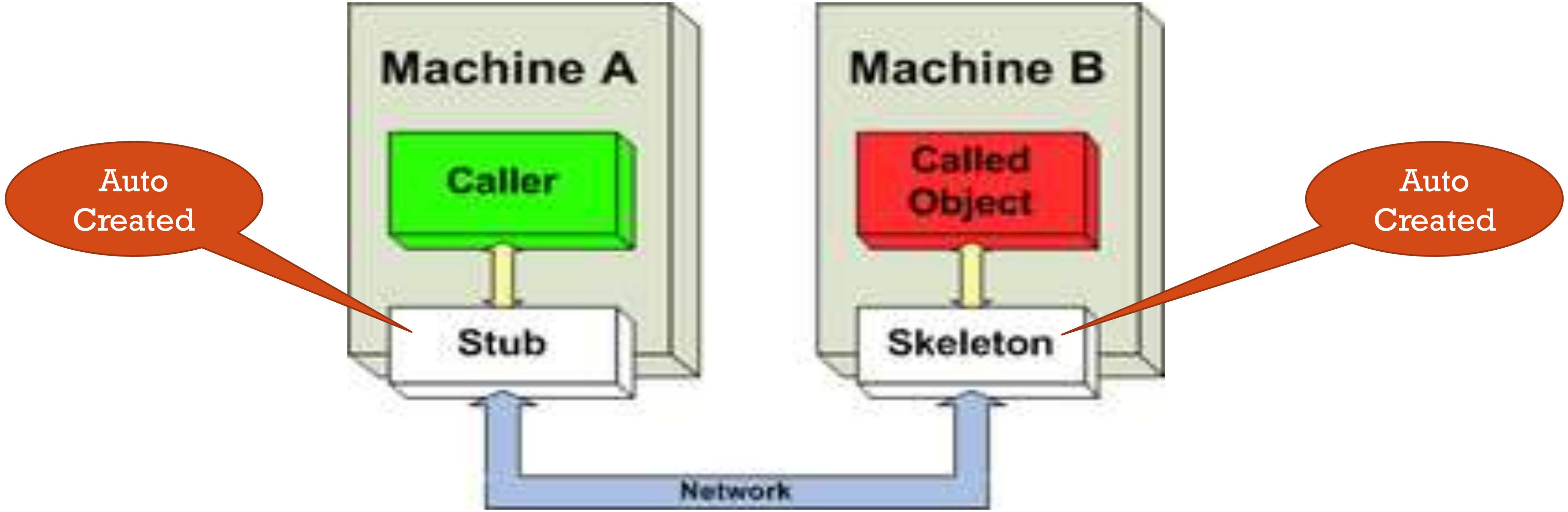
GENERAL ARCHITECTURE OF WEBSERVICES


SERVICE DEFINITION DOCUMENT
1. URL
2. Service calls and signatures
3. Accepted message format
4. Transport
Request/ Response
SERVICE CLIENT
Communication Channel / Transport (ex: HTTP, SMTP)
SERVICE SERVER

TYPICAL WSDL SOAP WEBSERVICES



WSDL COMMON USAGE



LETS SEE A LIVE EXAMPLE!

PROBLEMS WITH WSDL-SOAP

- XML can become very verbose
  - So there is a tradeoff between bandwidth, data transfer, etc
- Primarily for heavy duty work
  - Can transfer binary data through attachments
- XML is designed for machine readability
  - Tedious to debug
  - Even though tools are available
- You don’t need
  - A supercomputer for browsing facebook
  - A written contract and registrar for borrowing 10 Rs from a friend
- The necessity of webservices have become widespread
  - Unlike earlier we can imagine using webservices even for simpler things
  - The advent of smartphones have exacerbated this

REST TO THE RESCUE

- Piggybacks on HTTP verbs (we will soon see how)
- JSON format though not directly connected to REST, it goes hand-in-hand with the proliferation of RESTFul services
- JSON is a pleasure to deal with (Thanks Doug Crockford)
  - (read) Javascript the good parts
- Cons
  - No machine readable definitions (at least till a while back)
    - Remember I am old!
  - Hence, No auto *stub* generation tools
    - But we don't need them anyway
      - because its simple

REFERENCE

- Slide Outlines thanks to : https://www.cl.cam.ac.uk/~ib249/teaching/Lecture1.handout.pdf